УДК 519.71

О СТОЙКОЙ ОБФУСКАЦИИ КОМПЬЮТЕРНЫХ ПРОГРАММ

Н. П. ВАРНОВСКИЙ[1]
В. А. ЗАХАРОВ[1]
Н. Н. КУЗЮРИН[2]
А. В. ШОКУРОВ[2]

[1] *Московский государственный университет им. М.В. Ломоносова*

e-mail: zakh@cs.msu.su

[2] *Институт системного программирования Российской академии наук*

e-mail: nnkuz@ispras.ru

В данной работе предлагается схема аппроксимации функций и их производных по эмпирическим данным. Она основана на использовании известной из математического анализа формулы, позволяющей выразить дифференцируемую функцию через производную

Ключевые слова: интерполяция, оценка производной, частотное представление, устойчивость вычислений.

Введение

Обфускацией программ называется такое преобразованием программ, которое сохраняет все функциональные характеристики исходной программы, лишь незначительно ухудшает ее быстродействие, но при этом делает чрезвычайно трудоемким извлечение из открытого текста программы ключевой информации об устройстве содержащихся в ней алгоритмов и структур данных. Впервые концепция обфускации программ была предложена и подробно исследована в статье [1]. В этой и последующих работах обсуждалась возможность применения обфускации программ для защиты авторских прав на программное обеспечение [1], для предотвращения реинженерии программ [2], для создания и защиты «водяных знаков» [3], для информационной защиты мобильных агентов [4], для обеспечения приватности информационного поиска [5] и др. Обфускация программ также находит применение в криптографии; с ее помощью можно конструировать системы шифрования с открытым ключом [6-8], проводить вычисления над зашифрованными данными [6,9] и перешифрование сообщений [10], создавать верифицируемые системы тайного голосования [11] и др.

Однако методы обфускации программ имеют ценность лишь в том случае, если они сопровождаются строго обоснованными гарантиями их стойкости. Стойкость методов обфускации программ можно оценивать в зависимости от того, какие цели преследует противник, получивший доступ к тексту обфускированной программы, и какими ресурсами он располагает для достижения своей цели. Целью противника может быть получение информации об устройстве программы, вычисляемой ею функции, содержащихся в программе алгоритмах и структурах данных. Ресурсы, которыми располагает противник, включают предварительное знание об исходной программы, а также время, отведенное для решения задачи деобфускации, набор инструментальных средств анализа программ и др. Кроме того, возможности противника зависят от того, в какой форме представлена обфускированная программа (например, противник может иметь доступ ко всей программе или только к некоторой выделенной ее части). Эти три параметра (цель, предварительные знания и вычислительные ресурсы) полностью определяют модель противника.

В криптографии задачу взлома систем защиты, т. е. цель, стоящую перед противником, принято называть *угрозой безопасности* этой системы, а предположения о знаниях и возможностях противника, т. е. о его ресурсах – *атакой* на систему защиты информации. Традиционно классификация и оценка стойкости систем защиты

информации проводят относительно пар «угроза-атака», определяющих модель противника. Поэтому классификацию определений стойкости обфускации программ также наиболее естественно проводить относительно возможных моделей противника с учетом различных форм представления программ. В настоящей заметке для различных моделей противника приведены формальные определения стойкости обфускации программ, а также рассмотрены положительные и отрицательные результаты, оценивающие возможность построения стойких обфускаторов.

Стойкость обфускации программ

1. Модель «черного ящика». Впервые строгое математическое определение стойкости обфускации программ было предложено в работе [6]: обфускация считается стойкой, если всякий противник может извлечь из текста обфускированной программы за разумное (полиномиальное относительно размера входных данных) время не больше информации, чем можно было бы получить, проводя тестовые испытания этой программы как «черного ящика».

Определение 1. Вероятностный алгоритм O называется *обфускатором программ, стойким в модели «черного ящика»*, если он удовлетворяет следующим трем требованиям:

1. Для всякой машины Тьюринга (МТ) M любое выходное слово $O(M)$ машины O на входе M описывает МТ, вычисляющую ту же функцию, что и машина M (требование функциональности);

2. Длина описания и время выполнения любой МТ $O(M)$ не более чем полиномиальны от соответствующих параметров машины M (требование эффективности);

3. Для любой полиномиальной вероятностной МТ (PPT) A (противник) существует PPT S (симулятор) и пренебрежимо малая функция ν, удовлетворяющие для любой МТ M соотношению $|\Pr[A(O(M))=1]-\Pr[S^{M}(1^{|M|})=1]|\le \nu(|M|)$ (требование стойкости).

В статье [6] было установлено, что стойких обфускаторов программ в модели «черного ящика» не существует. Вместе с тем, в работах [8,9,10,13] было показано, что для отдельных классов функций (т.н. «точечных» функций) стойкая обфускация вычисляющих их программ возможна при тех или иных криптографических предположениях. Наиболее значительное продвижение было достигнуто в работе [10]; ее авторы доказали осуществимость стойкой обфускации программ перешифрования сообщений. В целом, однако, результаты исследований в этом направлении не дают больших оснований для оптимизма: до сих пор не удалось обнаружить достаточно сложной функции, программы вычисления которой допускают доказуемо стойкую обфускацию в модели «черного ящика».

2. Модель «серого ящика». В модели «черного ящика» противник стремится извлечь хотя бы какую-нибудь информацию об исходной программе. В модели «серого ящика» угроза противника ослаблена: предполагается, что трассы вычисления исходной программы не являются секретом для противника. Для формулировки требования стойкости обфускации в новой модели введем оракул $Tr(M)$, представляющий собой математическую модель «серого ящика». На входе x оракул $Tr(M)$ возвращает пару $(y, tr_{M(x)})$, где y – результат работы машины M на входе x, а $tr_{M(x)}$ – трасса вычисления машины M на этом входе. Строка $tr_{M(x)}$ определяется как конкатенация всех инструкций машины M, последовательно выполняемых на входе x.

Определение 2. Вероятностный алгоритм O называется *обфускатором программ, стойким в модели «серого ящика»*, если он удовлетворяет требованиям функциональности, эффективности, а также следующему требованию стойкости: для любой PPT A существует PPT S и пренебрежимо малая функция ν, удовлетворяющие для любой МТ M соотношению $|\Pr[A(O(M))=1]-\Pr[S^{Tr(M)}(1^{|M|})=1]|\le \nu(|M|)$.

Нам удалось установить, что универсальных обфускаторов, стойких, стойких в модели «серого ящика», также невозможно построить.

Теорема 1. *Эффективные обфускаторы, стойкие в модели «серого ящика», не существуют.*

Доказательство. Предположим, что существует эффективный алгоритм (PPT) $O(M)$, удовлетворяющий требованиям определения 2. Для каждой тройки $\alpha, \beta \in \{0,1\}^n$ и $b \in \{0,1\}$ определим машину Тьюринга $C_{\alpha,\beta,b}$ следующим образом. Пусть $x_1, x_2, \ldots$ – последовательность строк, подаваемых на вход машине $C_{\alpha,\beta,b}$. Если $x_1 = \alpha$, то машина выдает β, в ответ на $x_1 = \beta$ или $x_2 = \beta$ она выдает b, а во всех остальных случаях она выдает в ответ число 0.

Пусть r – случайная строка полиномиальной длины, используемая вероятностным алгоритмом O. Определим функцию $f(\alpha,\beta,b,r) = O(C_{\alpha,\beta,b,r})$. Очевидно, что функция f эффективно вычислима. Поскольку значение функции f однозначно определяет функцию, вычисляемую машиной $C_{\alpha,\beta,b}$, значение бита b также определено однозначно.

Машина S, имеющая оракульный доступ к функции, вычисляемой машиной $C_{\alpha,\beta,b}$, может получить в качестве ответа от оракула ненулевое значение лишь с пренебрежимо малой вероятностью. Отсюда $\Pr\{S^{C_{\alpha,\beta,b}} = b\} \le 1/2 + \nu(n)$, где ν – пренебрежимо малая функция. Следовательно, согласно определению 2, для любой полиномиальной вероятностной машины Тьюринга A $\Pr\{A(F(\alpha,\beta,b,r) = b\} \le 1/2 + \nu(n)$. Таким образом, b – трудный предикат функции f, и функция f односторонняя.

Теперь достаточно построить, используя одностороннюю функцию, семейство функций, необфускируемое в модели с "серым ящиком". Полученное противоречие докажет теорему.

Основная идея дальнейшего доказательства достаточно проста. Барак и др. [6] построили такое бесконечное семейство машин Тьюринга, что некоторый предикат $\pi(M)$, заданный на этом семействе, невыведываем при оракульном доступе к функции, вычисляемой машиной M, но может быть легко вычислен, если дан текст любой программы, эквивалентной M. Достаточно модифицировать эту конструкцию таким образом, чтобы следующие два требования выполнялись одновременно. Во-первых, выведываемость предиката $\pi(M)$ при наличии текста любой программы, эквивалентной M, должна сохраняться. Во-вторых, трассы выполнения машины M не должны давать противнику никакой дополнительной полезной информации, по сравнению с парами (вход, выход), получаемыми от «черного ящика». Противник, имеющий доступ к трассам выполнения машины M, представляется более сильным, чем противник, который видит только пары вход--выход. Для защиты от столь сильного противника мы используем криптографические примитивы. А именно, вместо сравнения входа x с фиксированной строкой α, как в работе Барака и др. [6], сначала проверяем равенство $f(x) = f(\alpha)$, где f – односторонняя функция. Лишь в том случае, когда равенство выполняется, мы проверяем равенство $x = \alpha$.

Типичная трасса $tr_{M(x)}$ состоит из инструкций, вычисляющих значение $f(x)$ и проверяющих равенство $f(x) = \alpha$. В подавляющем большинстве случаев результат проверки отрицательный. Более того, если равенство $f(x) = \alpha$ выполняется с вероятностью, которая не является пренебрежимо малой, то это противоречит тому, что функция f односторонняя.

Следует подчеркнуть, что до сих пор излагалась лишь общая идея доказательства. В действительности используется несколько более сложная конструкция.

Далее переходим к формальному доказательству. Контрпример, построенный Бараком и др. [6] использует два семейства машин Тьюринга. Для произвольной пары строк $\alpha, \beta \in \{0,1\}^n$ машина Тьюринга $C'_{\alpha,\beta}(x)$ выдает β, если $x = \alpha$, и выдает 0^n в противном случае. Для тех же параметров α, β машина Тьюринга $D'_{\alpha,\beta}(C)$ выдает 1, если $C(\alpha) = \beta$, и выдает 0 в противном случае.

Пусть $\{f : \{0,1\}^n \to \{0,1\}^n\}$ – односторонняя функция. Наша конструкция зависит также от целочисленного параметра $t \geq 2$, который может быть константой или значением произвольного, но фиксированного полинома (от n).

Выбираем случайно и равновероятно $2t$ строк $\alpha_1,\ldots,\alpha_t,\beta_1,\ldots,\beta_t \in \{0,1\}^n$. Обозначим эту совокупность из $2t$ строк через γ. Теперь определяем машину Тьюринга $C_\gamma(x)$ следующим образом. На входе x эта машина вычисляет $f(x)$ и сравнивает результат с заранее вычисленными значениями $f(\alpha_i)$, $i = 1,\ldots,t$. Если $f(x) \leq f(\alpha_i)$ для всех $i = 1,\ldots,t$, то C_γ выдает 0^n и останавливается. В случае, если $f(x) = f(\alpha_i)$ для некоторого i, C_γ проверяет равенство $x = \alpha$, и, если оно выполняется, выдает β, иначе выдает 0^n, и в обоих случаях останавливается.

Далее мы определяем машину Тьюринга D_γ. Она хранит два массива строк $\alpha_1,\ldots,\alpha_t$ и $\beta_1,\ldots,\beta_t$. Пусть i – индекс для обоих этих массивов, который изначально принимается равным 0. Получив на вход машину Тьюринга C, машина D_γ действует следующим образом.

1. Проверяет равенство $i=t-1$. Если оно выполняется, выдает 0 и останавливается.
2. Увеличивает индекс $i=i+1$ и затем вызывает машину C на входе α_i.
3. Если $C(\alpha_i) = \beta_i$, то проверяет текущее значение индекса. Если $i=t$, выдает 1 и останавливается, иначе выполняется переход на шаг 2.
4. Если $C(\alpha_i) \neq \beta_i$, то выдает 0 и останавливается.

Заметим, что моделирующая машина, имеющая доступ к машинам C_γ и D_γ как оракулам, может подать на вход машине D_γ произвольную машину C и извлечь значение α из трассы, затем подать машине C_γ в качестве запроса строку α_1 и получить β_1 (даже в случае модифицированной машины D_γ, скрывающей значения β_i) и т. д. По этой причине моделирующая машина может выдать лишь $t-1$ запросов машине D_γ (на самом деле, дальнейшие запросы допускаются, но завершаются нулевыми ответами, а потому могут игнорироваться). Необфускируемое свойство состоит в существовании t различных строк $\alpha_1,\ldots,\alpha_t \in \{0,1\}^n$ таких, что на каждой из них данная машина Тьюринга выдает ненулевой ответ.

Пара машин Тьюринга (C_γ, D_γ) заменяет машины $(C'_{\alpha,\beta}, D'_{\alpha,\beta})$, используемые в доказательстве отрицательного результата в работе Барака и др. [6]. Анализ этого доказательства показывает, что оно переносится на рассматриваемый случай, если верно следующее утверждение.

Утверждение. *Для любой полиномиальной вероятностной машины Тьюринга S выполняется равенство*

$$|\Pr\{S^{C_\gamma,D_\gamma}(1^n) = 1\} - \Pr\{S^{Z_\gamma,D_\gamma}(1^n) = 1\} = \nu(n),$$

где Z_γ *– машина, которая отличается от* C_γ *только тем, что на входе* α_t *выдает* 0^n*, и вероятности определяются случайным равновероятным выбором строк* $\alpha_1,\ldots,\alpha_t,\beta_1,\ldots,\beta_t \in \{0,1\}^n$ *и случайными величинами машины S.*

Доказательство. Достаточно показать, что любая полиномиальная вероятностная машина Тьюринга на любой из своих запросов к оракулу получает ненулевой ответ лишь с пренебрежимо малой вероятностью, вне зависимости от того, какая из двух машин, C_γ или Z_γ, используется в качестве первого оракула.

Для простоты изложения всюду далее предполагаем, что t=2. В этом случае машина S может выдать второму оракулу лишь один запрос. Пусть $a_1,\ldots,a_s \in \{0,1\}^n$ – все запросы машины S к первому оракулу до выдачи единственного запроса ко второму (подчеркнем, что s является случайной переменной). Предположим, что вероятность появления ненулевой строки среди ответов на эти s запросов не является пренебрежимо малой. Тогда можно построить полиномиальную вероятностную машину Тьюринга (обозначим ее чрез T) для инвертирования функции f.

Пусть $z \in_R \{0,1\}^n$ и $y=f(z)$ --- входная строка машины T. Машина T подбрасывает монету и, на основе исхода этого эксперимента, решает, какое из значений, $f(\alpha_1)$ или $f(\alpha_2)$ будет принято за y. Без ограничения общности, пусть это будет α_1. Далее, T выбирает случайные равновероятные строки $\alpha_2,\beta_1,\beta_2 \in \{0,1\}^n$ и вызывает S как подпрограмму. Машина T перехватывает все запросы к первому оракулу. Выполняя запрос $x \in \{0,1\}^n$, машина T вычисляет $f(x)$ и проверяет равенства $f(x)=y$ и $f(x)=f(\alpha_2)$. Если ни одно из них не выполняется, машина T выдает 0^n. В случае $f(x)=y$ машина T выдает β_1, а в случае $f(x)=f(\alpha_2)$ она действует так же как машина C_γ.

Очевидно, что вероятность получения ненулевого ответа от этого моделируемого оракула T не меньше, чем в случае реального оракула.

Поэтому для некоторого $j \in \{1,\ldots,s\}$ вероятность получить от машины T ненулевой ответ не является пренебрежимо малой. Для этого значения j, α_j принадлежит прообразу значения y с вероятностью 1/2. Таким образом, T инвертирует функцию f с вероятностью, которая не является пренебрежимо малой, что противоречит тому, что функция f является односторонней.

Если в ответах на первые s запросов ненулевая строка появляется лишь с пренебрежимо малой вероятностью, то вероятность получить ненулевой ответ на единственный запрос машины S ко второму оракулу также пренебрежимо мала.

Для остальных запросов к оракулу (после выполнения запроса ко второму оракулу) тот же аргумент, что и выше, показывает, что если вероятность успеха не является пренебрежимо малой, то существует эффективный алгоритм инвертирования функции f. Это противоречие доказывает утверждение, и, тем самым, завершает доказательство теоремы 1.

Ранее, в статье [14] нами было доказано, что если существуют односторонние функции, то обфускаторы, стойкие в модели «серого ящика», не существуют. Полученное усиление этого результата явилось следствием доказанного нами факта о том, что существование обфускаторов в смысле определения 2 влечет существование односторонних функций.

3. Обфускация с дополнительным входом. При проведении обфускации программ разумно предполагать, что противнику известны некоторые сведения о защищаемых программах. Например, противник может располагать инструкцией пользователя программой, каким-либо описанием функции, вычисляемой программой, или примерами обфускации некоторых программ. Естественно, эти дополнительные знания могут существенно усилить возможности противника. В криптогра-

фии атаки такого рода (например, атака с выбором зашифрованных сообщений) считаются наиболее сильными. В статье [7] было предложено формальное определение стойкости обфускации программ (в неуниформной модели вычислений) в случае, когда противник обладает дополнительными сведениями о защищаемых программах.

Определение 3. Вероятностный алгоритм O называется *обфускатором с дополнительным входом*, если он удовлетворяет требованиям функциональности, эффективности, а также следующему требованию стойкости: для любой PPT A существует такие PPT S и пренебрежимо малая функция ν, что для любого полинома $q(\cdot)$ и битовой строки z, длина которой не превосходит $q(|M|)$, выполняется соотношение $|\Pr[A(O(M),z)=1]-\Pr[S^M(1^{|M|},z)=1]|\le\nu(|M|)$.

В этой же работе было показано, что некоторые семейства функций не могут быть эффективно и стойко обфускированы в рамках определения 3.

4. Защита алгоритмов. Дальнейшее ослабление угрозы противника приводит нас к задаче применения обфускации для защиты алгоритмов. Предполагается, что противнику известна функция, вычисляемая программой, и его цель – извлечение информации об особенностях программной реализации этой функции. Знание противником функции, вычислимой программой M формально выражается предоставлением ему доступа к произвольной эквивалентной программе M_0, вычисляющей ту же самую функцию. Таким образом, мы приходим к следующему определению.

Определение 4. Вероятностный алгоритм O называется *обфускатором алгоритмов*, если он удовлетворяет требованиям функциональности, эффективности, а также следующему требованию стойкости: для любой PPT A существует PPT S и пренебрежимо малая функция ν, для которых соотношение $|\Pr[A(O(M),M_0)=1]-\Pr[S^M(1^{|M|},M_0)=1]|\le\nu(|M|)$ выполняется для любой пары эквивалентных МТ M,M_0.

Вопрос о существовании обфускаторов алгоритмов остается открытым. В следующей теореме мы представляем частичный отрицательный результат: такие обфускаторы не существуют в модели со случайным оракулом. В этой модели все алгоритмы, входящие в определение обфускатора, имеют доступ к оракулу, вычисляющему случайную функцию.

Теорема 2. В модели со случайным оракулом обфускаторы алгоритмов не существуют.

Доказательство. Определим дельта-функцию I_α, $\alpha\in\{0,1\}^n$, следующим образом: $I_\alpha(x)=1$, если $x=\alpha$, и $I_\alpha(x)=0$ во всех остальных случаях. Пусть $R:\{0,1\}^{2n}\to\{0,1\}^{2n}$ – функция, вычисляемая случайным оракулом.

Для всякой пары $\alpha,r\in\{0,1\}^n$ определим машину Тьюринга $M_{\alpha,r}$ следующим образом. Машина $M_{\alpha,r}$ хранит в своей памяти строки r и $\beta=R(\alpha,r)$. На входе $x\in\{0,1\}^n$ машина $M_{\alpha,r}$ обращается к случайному оракулу с запросом *(x,r)*. Если $R(x,r)=\beta$, то машина $M_{\alpha,r}$ выдает 1, во всех остальных случаях – 0, и останавливается.

Рассматривается семейство машин Тьюринга $\{M_{\alpha,r}\}$ для всех $\alpha\in\{0,1\}^n$ и всех длин входа n.

Согласно определению обфускатор сохраняет функциональность машины Тьюринга, и поэтому любая машина $O(M_{\alpha,r})$ вычисляет дельта-функцию I_α.

Рассмотрим следующего противника A. На входе $(O(M),\tilde{M})$ машина A находит n и далее вычисляет строку $r_0=r_0(n)$, $r_0\in\{0,1\}^n$. Здесь $r_0(\cdot)$ – произвольная эффективно вычислимая функция, которая каждому натуральному числу n ставит в со-

ответствие некоторую битовую строку длины n. Предположение о том, что машина A может найти значение параметра n не ограничительно, поскольку такую возможность всегда можно обеспечить с помощью специальным образом построенной машины $\widetilde{M}$.

Далее машина A выполняет машину $O(M)$, подавая ей на вход случайную строку $x \in_R \{0,1\}^n$. Если среди запросов машины $O(M)$ к оракулу встретится запрос (cx, r_0), то машина A выдает 1, в противном случае – 0.

Для дальнейшего зафиксируем какой-либо простой алгоритм вычисления дельта-функции I_α. Например, на входе $x \in \{0,1\}^n$ проверяем равенство $x = \alpha$, и если оно выполняется, выдаем 1, иначе – 0. Будем предполагать, что машина $\widetilde{M}_\alpha$ реализует именно этот алгоритм.

Сравним вероятность успеха противника в двух экспериментах. В первом из них машина A получает на вход обфускацию машины M_{α,r_0}. Очевидно, что

$$\Pr\{A(O(M_{\alpha,r_0}),\widetilde{M}_\alpha) = 1\} \geq 1 - \nu(n),$$

где ν – пренебрежимо малая функция. Во втором эксперименте машина A получает обфускацию машины $M_{\alpha,r}$ для случайной строки $r \in_R \{0,1\}^n$. В этом случае

$$\Pr\{A(O(M_{\alpha,r}),\widetilde{M}_\alpha) = 1\} \leq \nu(n).$$

Машина S получает только описание машины $\widetilde{M}_\alpha$ и оракульный доступ к дельта-функции I_α. Это не дает ей никакой информации о значении r_0. Следовательно, машина S не может различить два случая, соответствующих описанным выше экспериментам, и требования определения не могут быть выполнены. Теорема доказана.

Вместе с тем, обфускацию простейших алгоритмов можно провести сравнительно просто. В работах [15,16] показано, каким образом можно провести обфускацию детерминированных конечных автоматов и упорядоченных двоичных разрешающих диаграмм (OBDD).

5. Защита параметров. Во многих криптографических приложениях обфускация применяется лишь для одной цели – скрыть содержащуюся в программе константу (ключ), выбранную случайным образом из некоторого конечного множества. При этом считается, что сам алгоритм, реализуемый программой, известен противнику. Для формализации задачи обфускации программ в такой постановке рассмотрим параметризованное множество МТ $\mathrm{M} = \{M(c) : c \in \{0,1\}^n\}$, отличающихся друг от друга только константами c.

Определение 5. Вероятностный алгоритм O называется *обфускатором константы*, если он удовлетворяет требованиям функциональности, эффективности и следующему требованию стойкости: для любой PPT A существует PPT S и пренебрежимо малая функция ν, для которых неравенство $|\Pr[A(O(M(c)), M(c_0)) = 1] - \Pr[S^{M(c)}(1^{|M(c)|}, M(c_0)) = 1]| \leq \nu(n)$ выполняется для любой пары констант c, c_0, равномерно случайно выбранных из множества $\{0,1\}^n$.

Возможность построения стойких обфускаторов констант, зависит от параметризованного множества МТ M. Если $M(c)$ – это универсальная машина Тьюринга, в которой константа c играет роль моделируемой программы, то задача обфускации константы равносильна задаче обфускации программ в модели «черного ящика» с дополнительным входом. В статье [7] показано, что универсального стойкого обфускатора программ с дополнительным входом не существует. В то же время, алгоритм шифрования в криптосистеме с открытым ключом можно рассматривать как разно-

видность обфускации, скрывающей константу – секретный ключ. Обфускацию сингулярных функций, исследованную в работах [9, 12], также можно рассматривать как стойкую защиту констант.

6. Сокрытие предикатов. Наиболее слабая угроза – это распознавание некоторого функционального свойства (предиката) программы. Таким свойством может быть, например, наличие в программе скрытых (недекларированных) функциональных возможностей, побочных эффектов, вредоносных участков кода (вирусов).

Определение 6. Вероятностный алгоритм O называется *обфускатором предиката* π на множестве МТ M, если он удовлетворяет требованиям функциональности, эффективности, а также следующему требованию стойкости: для любой PPT A существует PPT S и пренебрежимо малая функция ν, удовлетворяющие для любой МТ M из класса M соотношению $|\Pr[A(O(M)) = \pi(M)] - \Pr[S^M(1^{|M|}) = \pi(M)]| \leq \nu(n)$.

Как видно из приведенного определения, обфускацию в модели «черного ящика» можно рассматривать как сокрытие всех возможных предикатов. Невозможность построения обфускатора, стойкого в модели «черного ящика», установленная в статье [6], не отрицает существования стойких обфускаторов, скрывающих отдельные предикаты для некоторых классов программ. Так, например, в статье [12] показано, что для класса программ, вычисляющих константы и точечные функции существует стойкая обфускация следующего свойства: «функция, вычисляемая программой, является константой».

Работа выполнена при поддержке гранта РФФИ 09-01-00632.

Литература

1. Collberg C., Thomborson C., Low D. A taxonomy of obfuscating transformations // Tech. Report, N 148, Univ. of Auckland, 1997.
2. Collberg C., Thomborson C., Low D. C. Manufacturing cheap, resilient and stealthy opaque constructs // Symp. on Principles of Programming Languages, 1998, p. 184-196.
3. Collberg C, Thomborson C. Watermarking, tamper-proofing, and obfuscation – tools for software protection // IEEE Transactions on Software Engineering, v. 28, N 6, 2002.
4. D'Anna L., Matt B., Reisse A., Van Vleck T. , Schwab S., LeBlanc P. Self-protecting mobile agents obfuscation report // Report #03-015, Network Associates Laboratories, 2003.
5. Ostrovsky R., Skeith III W.E. Private searching on streaming data // Lecture Notes in Computer Science, v. 3621, 2005, p. 223-240.
6. Barak B., Goldreich O., Impagliazzo R., Rudich S., Sahai A., Vedhan S., Yang K. On the (Im)possibility of obfuscating programs // Lecture Notes in Computer Science, v. 2139, 2001, p. 1-18.
7. Goldwasser S., Kalai T.Y. On the impossibility of obfuscation with auxiliary input // Proc. of the 46 th IEEE Symp. on Foundations of Computer Science, 2005, 553-562.
8. Hofheinz D., Malone-Lee J., Stam M. Obfuscation for cryptographic purposes // Lecture Notes in Computer Science, v. 4392, 2007, p. 214-232.
9. Lynn B., Prabhakaran M., Sahai A. Positive results and techniques for obfuscation // Lecture Notes in Computer Science, v. 3027, 2004, p. 20-39.
10. Hohenberger S., Rothblum G. N., Shelat A., Vaikuntanathan V. Securely obfuscating reencryption // Lecture Notes in Computer Science, v. 4392, 2007, p. 233-252.
11. Adida B., Wikstrom D. Obfuscated ciphertext mixing // IACR. Eprint Archive, N 394, 2005.
12. Wee H., On obfuscating point functions // Proc. of 37th ACM Symp. on Theory of Computing, 2005, p. 523-532.
13. Varnovsky N.P., Zakharov V.A. On the possibility of provably secure obfuscating programs // Lecture Notes in Computer Science, v. 2890, 2003, p. 91-102.
14. Varnovsky N.P. A note on the concept of obfuscation // Труды Института системного программирования, т. 6, 2004.
15. Goldwasser, S., Rothblum G. On best-possible obfuscation // Lecture Notes in Computer Science, v. 4392, 2007, p. 253-272.
16. Kuzurin N.N., Shokurov A.V., Varnovsky N.P., Zakharov V.A. On the concept of software obfuscation in computer security // Lecture Notes in Computer Science, v. 4779, 2007, p. 281-298.

ON THE SECURE OBFUSCATION OF COMPUTER PROGRAMS

N. P. VARNOVSKY[1]
V. A. ZAKHAROV[1]
N. N. KUZURIN[2]
A. V. SHOKUROV[2]

[1] *Lomonosov Moscow State University*

e-mail: zakh@ cs.msu.su

[2] *Institute for System Programming of Russian Academy of Science*

e-mail: nnkuz@ ispras.ru

The aim of program obfuscation is to bring a program into such a form, which impedes the understanding of its algorithm and data structures or prevents extracting of some valuable information from the text of a program. In this paper we address the issue of defining security of program obfuscation. We analyze several formal definitions of obfuscation security, consider positive and negative results on obfuscation security and the applications where these model may be valid.

Keywords: computer security, program obfuscation, security, Turing machine, cryptosystem.